Computational Visual Media
https://doi.org/10.1007/s41095-018-0124-x Vol. 4, No. 4, December 2018, 349–365

Research Article

Resolving display shape dependence issues on tabletops

James McNaughton[1] (✉), Tom Crick[2], and Shamus Smith[3]

Abstract Advances in display technologies are transforming the capabilities—and potential applications—of system interfaces. Previously, the overwhelming majority of systems have utilised rectangular displays; this may soon change with digital devices increasingly designed to be ubiquitous and pervasive, to facilitate frictionless human interaction. At present, software is invariably designed assuming it will be used with a display of a specific shape; however, there is an emerging demand for systems built around interacting with tabletop interfaces to be capable of handling a wide range of potential display shapes. In this paper, the design of software for use on a range of differently shaped tabletop displays is considered, proposing a novel but extensible technique that can be used to minimise the influence of the issues of using different display shapes. Furthermore, we present a study that applies the technique to adapt several software applications to several different display shapes.

Keywords visual content management; irregular displays; screen design; multi-touch surfaces; tabletop displays; ubiquitous computing

1 Introduction

The majority of tabletop software systems that provide visual feedback to a user are designed to do so with a specific display shape, with the most common of these shapes being rectangular. However, adapting displays to different shapes has always been possible by covering regions of a display [1]. This, in addition to new technologies such as circular liquid crystal displays [2, 3], allows the potential for different display shapes to be used in tabletop systems. With the wide application and deployment of digital technologies, developers increasingly need to consider how to make their software and systems agile enough to adapt to not only displays of varying sizes and aspect ratios, but also to displays of varying shape. In support of this aim, this paper presents a novel technique that allows developers to adapt tabletop software to different display shapes, as well as a study evaluating its potential.

2 Background

Traditional displays used in digital systems are overwhelmingly rectangular. However, an increasing number of non-rectangular displays is becoming available for a variety of real-world applications, with patents appearing from the early 2000s [4]. One such instance of a non-rectangular display is Toshiba's circular thin film transistor liquid crystal display (LCD) [2]; this display is typically employed in vehicles to show dashboard and operating information. The display is designed to look similar to a typical dial on a car dashboard, thus requiring the display to be circular. The Motorola Aura [3] is a mobile phone that showcases another example of a non-rectangular (circular) display.

A non-rectangular display is utilised in the PyMT—a multi-touch framework for Python [5]—project. A circular display is made by projecting the system's output onto a circular surface. The Puddle of Life application is designed to tailor its visual output to this circular display shape; the software is constrained by the specifications of the hardware. This is also

1 School of Education, Durham University, Durham DH1 3LE, UK. E-mail: j.a.mcnaughton@durham.ac.uk (✉).
2 Department of Computer Science, Swansea University, Swansea SA2 8PP, UK. E-mail: thomas.crick@swansea.ac.uk.
3 School of Electrical Engineering & Computing, University of Newcastle, Callaghan NSW 2308, Australia. E-mail: shamus.smith@newcastle.edu.au.

Manuscript received: 2018-03-12; accepted: 2018-08-31

true for software built for circular dashboards [2] and phone [3] displays. As a result, such software cannot be used easily with other non-traditional display shapes, and so applications used with these circular displays are not suitable for use with typical rectangular displays.

The DiamondSpin framework [6] offers support for circular tabletop multi-touch displays. Instances of a circular display used by the framework are achieved by either projecting the system's visual output onto a circular surface or occluding sections of the output. DiamondSpin supports these displays through several applications which are designed specifically for use on a circular tabletop interface; the framework provides several features to support applications intended for circular interfaces, such as the ability to orientate items towards the nearest point on the edge of the display. Some of the example applications for this framework can be used with both rectangular and circular interfaces; using the framework's features, the applications are able to appropriately arrange the layout of content to the display shape used. However, this adaptive ability is limited to two regular display shapes: rectangular and circular.

The circular dashboard LCD [2] is a typical example of a technology designed with the intention of supporting ubiquitous computing [7]. We have entered an era of ubiquitous and pervasive digital systems designed to fit naturally into their surroundings [8], recognising the potential impact on a user's emotion and behaviour [9]. However, for this to be natural and intuitive, systems must be designed around emerging user environments, not the status quo. The implication is that many previous standard elements of typical computing systems need to be reconsidered, such as the shape of a system's interface. One of the strengths of tabletop systems is their ubiquitous nature [10]; the ability for a tabletop system to manage different display shapes enhances this strength, allowing interfaces to better fit their environment.

Previous work that discusses the possible effects of displays with non-rectangular display shapes [11] highlights the benefits it may offer; these benefits include potential improvements to collaboration around the display between users. Vernier et al. [11] proposed a circular tabletop interface which would allow each user to have an equal share of the display. Though focused on circular tabletop displays, they highlight the effect that a different display shape could have on the use of an interface; benefits noted in the work include the improved management of each user's personal space.

It is thus evident that the visual content of systems will need to accommodate the use of different display shapes. A natural way to achieve this is to define specific layouts for the software's visual content for each display shape it may potentially be used with. This is the approach used in the software systems discussed thus far [5, 6]; however, for software which may have a wide range of potential interface shapes, this would require significant extra development work.

The structured literature review [12] that informed this research project highlighted the past lack of innovation in utilising different display shapes. This is likely due to the cyclical nature of dependency between display technology and its supporting software. However, research activity in this area is increasing, with recent research detailing investigations on adapting text to non-rectangular displays [13] and how visual content is best presented on legacy interfaces [14]. The outcomes from these two projects are a set of guidelines to be followed to ensure content can fit different display shapes; however, these guidelines can be restrictive and place additional responsibilities on software developers and content designers. This, much like designing for unique content configurations for specific display shapes, will result in potentially significant additional development and design work.

A potential method to reduce this would be to use a tool which can adapt the layout of visual contents to different platforms, such as GUMMY [15]. This tool can take an existing layout as defined by a developer and quickly adapt it to the parameters of a specific platform, including the restrictions of the display. For example, the spacing of a predefined layout may be reduced by the tool when the target platform is known to use a small display. By implementing a method which allows the tool to dynamically adapt a layout of content items to different display shapes, software developers could produce multiple layouts for a piece of software to use with different display shapes relatively easily.

Instead of using predefined layouts for each potential display shape, software could be designed to use a single layout which is adapted to fit different display shapes automatically. For example, SUPPLE [16] is a system which adapts the layout of contents to fit the parameters of a display, with the content automatically arranged by the system to fit various display sizes and resolutions. Furthermore, SUPPLE has the ability to adapt its visual contents based on input type. However, the system assumes that despite the changes in an interface's size, resolution, input type, and colour support, it will always be rectangular. Significant changes to layout adapting systems, such as SUPPLE, will be needed to allow them to work with non-rectangular displays.

Preliminary research into software capable of adapting its visual contents to different display shapes also exists. Waldner et al. [17] discuss work carried out on the development of a system which allows windows to be adapted to make use of unusual display shapes. The unusual display shapes presented by Waldner et al. consist of a series of overlapping projections which form the system's output, with these overlapping projections not always forming a rectangular shape. Therefore, it is important for the software used not to be dependent on a rectangular visual output. The technique presented works by automatically identifying the best location for new windows. An "importance" value is used for each pixel of the output and the magnitude of this value is used to assess its suitability to displaying a new window. The higher the value is, the less suitable the pixel is; pixels outside the output's shape are given a maximal importance value. This means that no window can be placed anywhere which will result in it occupying these pixels. This prevents content windows being placed where they may be partially occluded due to some of their regions existing outside the display area, ensuring that no visual content is obscured.

Waldner et al.'s technique appears to be beneficial for use with windows which have no relative placement defined with respect to each other. The technique could thus be employed in other systems to manage the placement of visual content items when used with different display shapes. However, for visual content which may have a significant locational relation this technique may be unsuitable; for example, the order in which windows are created will influence where the technique positions them. The technique currently has no method to accommodate any locational relationships which should exist between windows. When two visual content items must be in close proximity to each other, there is no guarantee that the technique will position them together. As the positioning of content items can imply functional relations to the user [18], the loss of intended locational relations is undesirable for some systems.

The potential benefits [8, 11] and growing supporting technologies [2, 3] of non-rectangular displays indicate an increasing demand for systems to support them; more specifically, there is a growing demand for different display shapes in tabletop systems [5, 6]. For tools which are used to design the layout of software [15] or systems which dynamically update the layout of visual content items [16], a new method of adapting visual content layouts to different tabletop display shapes is needed.

3 Identifying display shape dependence issues

To develop a method of adapting visual content layouts to different display shapes, the effects of applying a new display shape to a system's visual output need to be considered. Issues can occur when a display shape is used which differs from the shape of a display assumed by the software; in this section we discuss six of these *display shape dependence issues* (*DSDIs*).

The DSDIs presented in this work are entirely novel; they were derived from first-hand observations of attempts by the authors to manually fit visual contents to different display shapes, and from observations discussed in related work of Serrano et al. [13, 14]. Figure 1 puts the DSDIs into two groups: primary and secondary. *Primary DSDIs* are issues which can occur when displaying content intended for a particular display shape on a display of another shape, without modification. *Secondary DSDIs* are issues which can occur when performing modifications when displaying content intended for a particular display shape on a display of another shape, e.g., when attempting to resolve any primary DSDI. Note that the order of the DSDIs given here has no significance and is only used for identification purposes.

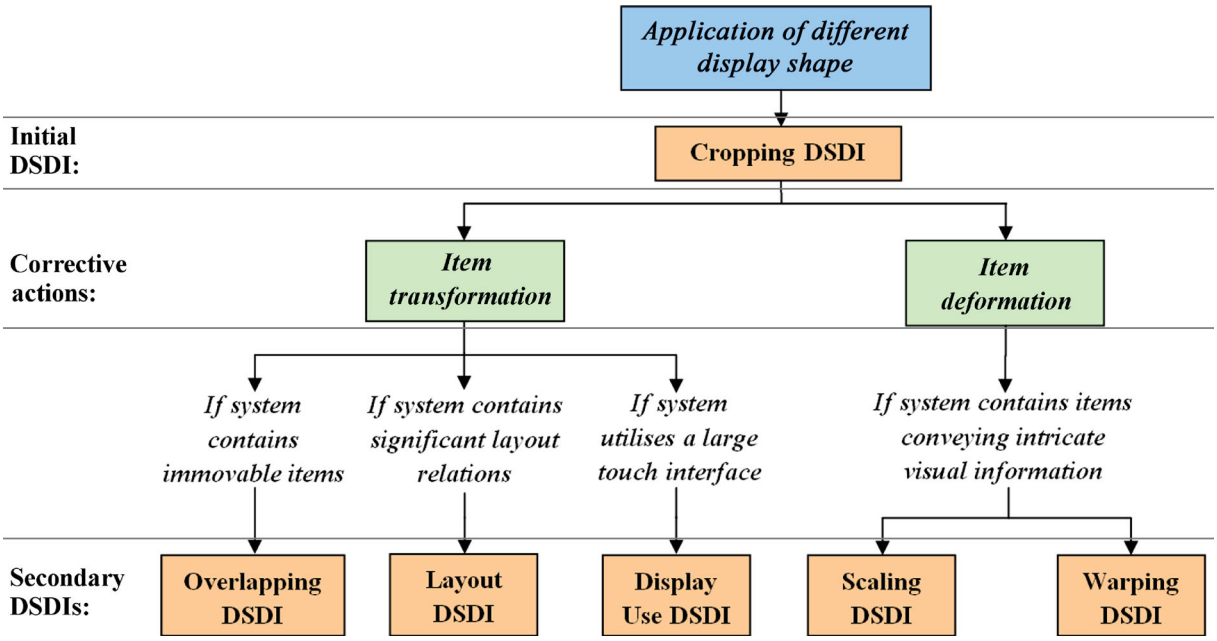

Fig. 1 A flowchart indicating causality of display shape dependence issues (DSDIs).

i) ***Cropping DSDI:*** *Content items appear cropped, as parts of them are positioned outside the display area.*

The scope of DSDIs outlined in this work is limited to environments where there is no hardware-level attempt to fit the visual contents to the display shape. This is in keeping with the ideas in Section 2 where the majority of works considered create non-rectangular displays by showing a region of a rectangular output by, e.g., partially covering parts of a display or occluding areas of a projection.

Cropping can make visual content items unfit for purpose if visual information they convey is lost. Users may also be unable to interact directly with cropped items. The cropping of visual content items outside the display area was also noted by Waldner et al. [17]. One method of resolving cropping issues is to make software capable of adapting to different display shapes. Attempts to resolve cropping can result in secondary DSDIs.

ii) ***Overlapping DSDI:*** *Occlusion by other visual content items.*

Attempts to resolve the *Cropping DSDI* by movement of content items can result in further issues such as causing items which did not previously overlap to cover each other. This can result in regions of visual content items becoming obscured, with the same negative impact as the *Cropping DSDI*. Overlapping content items is not always an issue, especially if the obscuring items can be moved by the user, but in systems where these items are immovable the occlusion may hide important information.

iii) ***Layout DSDI:*** *Loss of layout relations between visual content items.*

Changing content item positions may be an issue in applications where layout is important, as the positioning of content items can imply functional relations between them to the user [18, 19]. For example, controls in an application may be placed next to the items they influence. If this layout relationships are lost, this may confuse users as to what items the set of controls influences. The loss of layout may not make content unfit for purpose, but could be highly undesirable for software where the layout is intended to aid the user. However, loss of layout is not always problematic, and this DSDI only applies to specific content, tasks, or applications.

iv) ***Scaling DSDI:*** *Excessive scaling of visual content items.*

Scaling—resizing a visual element whilst maintaining its aspect ratio—may be performed to avoid cropping content. If this scaling is excessive, the visual information displayed by a content item may no longer be communicated. For example, if a textual content

item is excessively scaled down, its characters may become too small to be legible to users.

v) ***Warping DSDI:*** *Warping of visual content items.*

Warping occurs when the contents of a system are stretched and/or squashed in a non-uniform manner [20]. This may be applied to the entire visual output of a system to make it fit a particular display shape, causing content items to be stretched and squashed non-uniformly. Even when such deformation is relatively small, it can make visual information in a content item incomprehensible to the user. This DSDI may not apply in all circumstances; some visual content items may display visual information which is comprehensible to the user no matter how severely it is deformed. For example, if a content item's visual information is its colour, then as long as the item can be seen, it can successfully communicate its visual information to the user.

vi) ***Display Use DSDI:*** *Display regions remain unused by the layout of visual content items.*

As a result of attempting to resolve cropping, visual content items may often be translated to the same regions of the display, leaving large areas of an interface unused. While this may not make the software unfit for purpose, it is not desirable for large tabletop interfaces, as on larger interfaces some regions of the display may be beyond a user's reach. If the display uses direct touch interaction, items in these regions are isolated from the user's influence. There are solutions to this problem, such as users changing their positions around the interface or controls which can be used to indirectly manipulate out of reach items [21]. However, if all content items are constrained to a single region of the display beyond a user's reach, the need to repeatedly use such solutions is undesirable. Changing positions to access remote content may not be possible in some environments due to the placement of the interface: for example, multi-touch interfaces may have several users positioned around them. Users interacting simultaneously with a single interface often have a tendency to establish "territories" [22]. A user wishing to move when using such interfaces may disrupt others who are also interacting with the system. Solutions involving the remote control of out of reach content items result in users manipulating content items through a proxy [23]. This means that their interaction is no longer direct, which can be undesirable as it counteracts the benefits of direct touch interaction [24]. Ensuring content items are spread out across the interface reduces the likelihood that all items are placed beyond the reach of the users.

For a tabletop software system to be capable of dynamically adapting its visual content to different display shapes, the *Cropping DSDI* needs to be resolved in a manner that minimises the influence of any potential secondary DSDIs. Figure 1 maps the causality of the DSDIs and notes circumstances in which they may have undesirable consequences on a system's visual content. In the next section, we use this list of DSDIs to present a technique to adapt software to different display shapes in a way that reduces the possibility of software becoming unusable due to these issues arising.

4 Minimising display shape dependence issues

We have developed a novel technique to resolve the *Cropping DSDI* while minimising the effect of secondary DSDIs. This technique is a method of adapting visual content items dynamically to different display shapes, using two stages. The first stage relies on taking the original visual output of the software and transforming it to fit within the borders of a new display shape. This stage is referred to as *virtual projection*. The second stage uses the difference between the shape of the virtual projection and the display shape to ensure that the layout can, if needed, utilise all regions of the display. This stage is referred to as *position pulling*.

4.1 Virtual projection

For the first stage, the original shape of the software environment's visual output is treated as a single item. This item, referred to as the virtual projection, is translated, rotated, and scaled so that it fits within the display shape being used. The transformations applied to the virtual projection are applied to all visual content items displayed in the software's original layout.

This stage of the technique has the effect of keeping the visual content items aligned with the virtual projection. This ensures that visual contents do not become occluded by lying outside the borders

of the new display shape: the content items are all contained within the virtual environment which fits inside the display. Figure 2 demonstrates how the virtual projection and its contents are transformed together to fit a new display shape.

As the original visual output of most current software is likely to be rectangular, the virtual projection for most systems can be assumed to be a rectangle. Finding the largest rectangle which can be placed within the display is required for this technique to utilise as much of the display as possible. The concept of finding the largest rectangle inside a shape is known as the *largest empty rectangle problem* or the *maximal rectangle problem*; there are several existing solutions to this problem [25, 26].

Methods also exist for finding the largest polygon of any specific shape within in the display shape [27], allowing for the adaptation of any software environment whose display was not originally rectangular. Therefore the technique can be used to adapt software originally designed for any specific display shape to other display shapes.

As all content items are translated, rotated, and scaled identically, their layout relations are preserved preventing the *Overlapping DSDI*. As no warping deformations are used, the *Warping DSDI* cannot occur. It is important to note that because the scope of this work is limited to tabletop displays, rotation of the content is not considered an issue. By definition, tabletop interfaces are horizontal, allowing them to be accessed from any direction, making orientation unimportant.

However, the *Scaling DSDI* may arise: if the virtual projection is significantly smaller than the initial layout's environment, the contained content items may be excessively scaled. Indeed, it is possible that the content items may be scaled to an extent that they become unfit for purpose. To counter this, the virtual projection technique adheres to scaling limits.

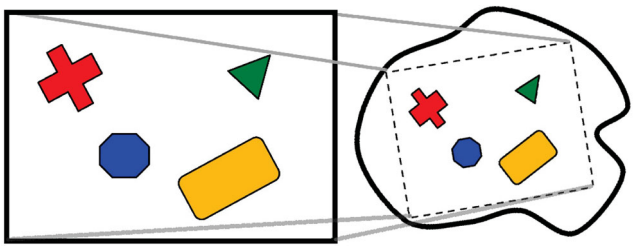

Fig. 2 An example of the *virtual projection* stage.

For systems where content items cannot be scaled to extremes, its content items should only be scaled within a predetermined limit. This may result in sections of the content items becoming occluded by mutual overlap or by the display edge if they cannot be made small enough. Without scaling limits, the *Cropping* and *Overlapping DSDIs* are guaranteed not to occur but the *Scaling DSDI* remains a risk; with scale limits, the potential risks of DSDIs are reversed. When utilising this technique, developers must decide on DSDI trade-offs in their software.

This stage of the technique is similar to previously presented methods of dynamically adapting content to different displays which entails constricting the placement of content to a specific region of the displays used [28, 29]. This region is recreated for every display shape using the same dimensions, ensuring the contained content keeps layout, scale, and orientation relative to the region.

This stage of the technique has the potential to minimise the effects of most DSDIs. However, it does not prevent occurrence of the *Display Use DSDI* since only a single region of the display, the virtual projection, is used.

4.2 Position pulling

The second stage of the technique fills the gaps between the virtual projection and the display edges. The objective of this stage is to move content items into areas of the display which are left unused by the virtual projection. The edges of the virtual projection exert a "pulling force" on content items. The magnitude of this force is determined by the sizes of the gaps between the display border and the virtual projection's edges. The magnitude of this pulling force is also influenced by a content item's proximity to the edge instigating the force. Content items are pulled from their centroids to simplify calculations: shapes and sizes of the items moved are ignored.

The position pulling stage moves content items into unused regions of the display (see Fig. 3). Items positioned near large gaps between the virtual projection and the display borders are pulled into these areas. Figure 3(left) shows the pulling forces which result from the gaps between the virtual projection area's edges adjacent to the content items. Figure 3(right) shows the translations of the content items resulting from the combined influence of the pulling forces.

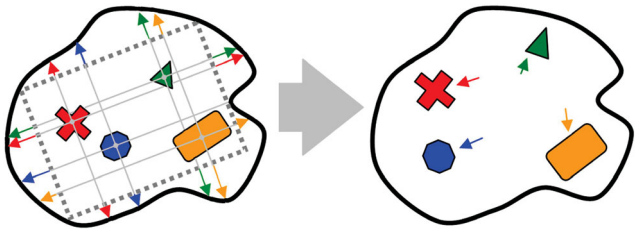

Fig. 3 An example of the *position pulling* stage.

Algorithm 1 shows how each vector representing the individual pulling force is calculated and made proportional to the item's proximity to the corresponding edge. Algorithm 2 details how pulling force vectors are computed; θ represents the orientation of the virtual projection. The notations in Algorithms 1 and 2 correspond to that in Fig. 4.

Algorithm 1 Calculating pulling force vectors

getPullVectors$(O,\ A_1,\ A_2,\ B_1,\ B_2,\ C_1,\ C_2,\ D_1,\ D_2)$
$\overrightarrow{LEFT} \leftarrow (\overrightarrow{D_1D_2} * (|\overrightarrow{OD_1}| \ / \ |\overrightarrow{B_1D_1}|))$
$\overrightarrow{RIGHT} \leftarrow (\overrightarrow{B_1B_2} * (|\overrightarrow{OB_1}| \ / \ |\overrightarrow{B_1D_1}|))$
$\overrightarrow{UP} \leftarrow (\overrightarrow{A_1A_2} * (|\overrightarrow{OC_1}| \ / \ |\overrightarrow{A_1C_1}|))$
$\overrightarrow{DOWN} \leftarrow (\overrightarrow{C_1C_2} * (|\overrightarrow{OA_1}| \ / \ |\overrightarrow{A_1C_1}|))$
return $\overrightarrow{LEFT},\ \overrightarrow{RIGHT},\ \overrightarrow{UP},\ \overrightarrow{DOWN}$

Algorithm 2 Applying the "pulling force" vectors

applyPull$(item,\ \theta,\ \overrightarrow{LEFT},\ \overrightarrow{RIGHT},\ \overrightarrow{UP},\ \overrightarrow{DOWN})$
$\overrightarrow{PULL} \leftarrow \overrightarrow{LEFT} + \overrightarrow{RIGHT} + \overrightarrow{UP} + \overrightarrow{DOWN}$
$\overrightarrow{PULL}.rotateBy(\theta)$
$item.translateBy(\overrightarrow{PULL})$
return

This stage of the technique effectively deforms the layout of the content items by warping it to fit the display shape. Although the layout is warped, the content items themselves are not deformed in any way by this stage of the technique. The deformation of the layout ensures that more of the display's real estate is utilised and that the layout is not constrained to a small region of the display. This minimises possible effects of the *Display Use DSDI*. The shape of each item does not influence the pull vectors, only the point it uses for translation. Taking shapes into account would require significantly more processing but would only result in a marginally more accurate pull vector.

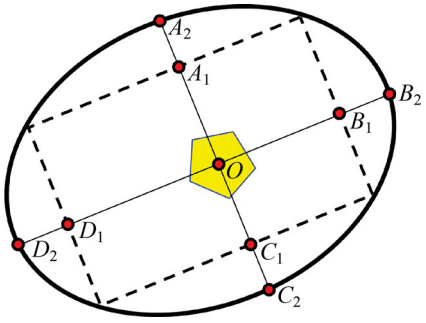

Fig. 4 Detailed view of the information used in *position pulling*.

The pulling forces given here assume the original environment of a piece of software is rectangular. This can be considered acceptable due to the current prevalence of software intended for use with rectangular displays [30]. However, it is possible that, in future, the technique may need modification to adapt software for non-rectangular displays to different shapes. The virtual projection stage can easily be modified to use the maximal polygon opposed to the maximal rectangle. For the position pulling stage, a vector representing the pulling force for every unique edge of the virtual projection would need to be created. Addition of these vectors would then create an overall vector which used to translate a content item.

4.3 Technique observations

Our method employing the two stages discussed has the potential to minimise any of the DSDIs identified, when used on tabletop displays, two-dimensional displays which are positioned horizontally. The technique allows for interventions at different points which can depend on how content should be treated, giving developers greater control over the influence and impact of the technique. In addition, the technique uses only linear transformations, resulting in a simple implement piece of functionality. These characteristics make our technique a promising approach to handling different display shapes on tabletops for real-world applications.

5 Study

A study was carried out in which an implementation of our technique was used for several different software applications, and several different display shapes.

5.1 Implementation of the technique

The technique was implemented within version 2.1 of the *SynergyNet* framework [31, 32], a multi-touch software framework intended for use with rectangular

tabletop interfaces. The framework functions by rendering a range of content items, such as images, text, and video, which can then be manipulated by users. *SynergyNet* is written in Java, allowing it to be run across different operating systems. It can accept a wide range of different multi-touch protocols such as TUIO [33]. The framework's ability to adapt to different systems makes it a suitable candidate for use with different display shapes.

Another reason for the choice of the *SynergyNet* framework was the range of readily available applications written using it. Implementing our technique within this framework allowed any application written for this version of *SynergyNet* to potentially utilise different display shapes. The available applications were produced by a number of different developers for different purposes, allowing the technique to be tried with various kinds of content, with not just differing layouts, but also differing significance to their layouts.

For the technique to work, the software needs to be informed of the dimensions of the display shape currently available. This is done via files containing vectors corresponding to the display shape. These vector files also include information on the display shape's maximal polygon (a rectangle is used as all previous *SynergyNet* applications assume a rectangular display).

Figure 5 shows the options available to application developers utilising the technique. Choices made in its usage, such as which stages of the technique are employed and the constraints placed on these stages, can influence which DSDIs it prevents and counters. For example, if a developer wishes a content item to act as the background to the entire display, and cropping it is not an issue, then both stages of the technique can be omitted. For content items other than background images, developers can choose whether or not to apply the position pulling stage of the technique. If applied, the content items maximise use of the display shape's real estate by being pulled into previously unused areas of the display. However, if the layout of the content items is of greater importance than the maximisation of display usage, the developer can omit the position pulling stage for certain content items in the layout.

5.2 Selection of applications

Our technique was tested using a number of applications created by several developers for previous studies in the *SynergyNet* project [34]. The study focused on two specific applications, *Grid* and *Simple Map* (see Fig. 6):

- ***Grid***:
  - Nine items are positioned in a grid layout which users can reposition, rotate, and scale through

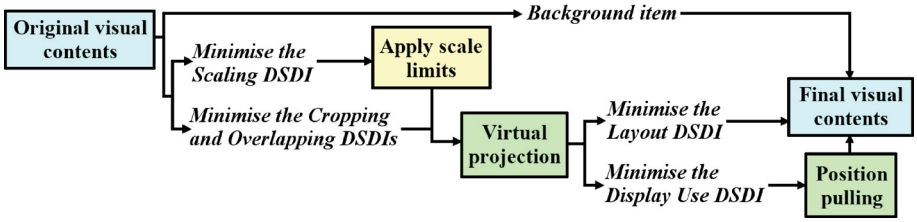

Fig. 5 Flowchart demonstrating navigation of the technique.

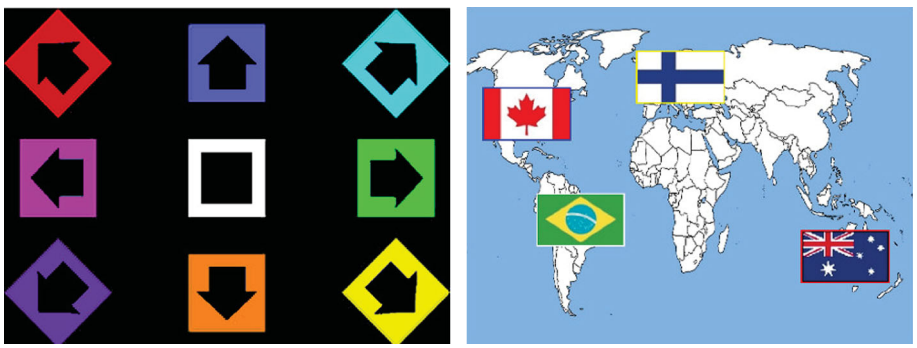

Fig. 6 Applications chosen for observation, from left to right: *Grid* and *Simple Map*.

multi-touch gestures;
  - All content in this application is specifically positioned;
  - Relative locations of all items in this application are significant;
  - This is one of the applications provided by the *SynergyNet* framework to help familiarise users with interacting with a large multi-touch interface.
- ***Simple Map***:
  - Users supply countries in a configuration file and the application shows a world map with the specified countries' flags positioned on top of them;
  - Users may move the flags around the map;
  - All content in this application is specifically positioned;
  - The locational relations between the flag items and the background image is strong.

These two were chosen because of the differing significance of their items' layouts, as the study intended to investigate how the technique would impact the arrangement of items having significant layout intent.

5.3 Selection of display shapes

The *SynergyNet* framework was adapted to allow use of a range of display shapes. These shapes were chosen after consultation with a UK-based furniture manufacturer with experience of designing multi-touch tables. The manufacturer was in a strong industry position for making predictions concerning the likely design features of future multi-touch tabletops, including their display shapes; the shapes chosen are shown in Fig. 7.

Some of these shapes were derived from common table shapes which could be used with tabletop interfaces, such as the semi-circular design. They allow users positioned around the display to have an equal share of the interface, which can be beneficial to collaboration. A circle was also chosen for its ability to allow users to have equal shares of the display. A triangle was chosen because of its regular nature: all of its sides and corners are congruent. The intersecting circles shape was designed to make further use of multi-touch's facilitation of collaboration by providing focal points and areas to share content items in. The larger circle of the display shape can be used for one activity such as displaying content items whereas the smaller circle could be used for another activity such as manipulating the items. A semi-circle was selected as it is a commonly used shape for tables.

5.4 Approach

In the study, both stages of the technique were always applied to all content items to investigate the full impact of the technique. No limitations were placed on the technique in the study. The technique was implemented in such a way that it was applied to the initial layout of content in an application.

The four different display shapes were emulated in the study by using a large rectangular display shape, then covering parts of the screen used to make the visible area of the shape match an intended display shape. Both of the applications of the study were used with each of the four display shapes. The framework was set up to capture screenshots taken at various times during application of the technique: (i) before application; (ii) after the placement of the display shape's outline (blanking out anything that would not visible on the display); (iii) after the virtual projection stage; and (iv) after the position pulling stage.

5.5 Data analysis

The impact of the technique on the DSDIs was measured using the screenshots taken during the execution of the technique. For each combination of the two software applications and four display shapes, the presence of DSDIs can be ascertained.

Fig. 7 Display shapes chosen for observation. Left to right: *circle*, *triangle*, *intersecting circles*, and *semi-circle*.

For each DSDI a measurement was outlined which could be obtained from the screenshots. These measurements were then used to evaluate the impact of the various stages of the technique on the DSDIs. The measurements were converted to percentages to allow simple comparison between the applications, as the sizes of their layouts, initial scales, and number of items differ. These measurements were as follows:

- **Occluded items**: To measure impact of the Cropping DSDI, we measured the percentage of items suffering occlusion by the display shape.
- **Overlapping items**: To measure impact of the Overlapping DSDI, the percentage of items suffering occlusion by other items was measured.
- **Change in layout**: To measure impact of the Layout DSDI, the distance of items from where they were expected to be with a layout of the same scale and orientation (but no other deformations) was determined. This average of these values, expressed as a percentage of the width of the layout's minimal bounding rectangle, was used.
- **Display shape unused**: To measure impact of the Display Use DSDI, a minimal bounding rectangle was drawn around the visual content. The area unused (i.e., area of the display outside the rectangle) divided by the total area of the display shape gives the fractional unused area.
- **Change in scale**: The impact of the Scaling DSDI was measured as the absolute maximum percentage change in scale of all items in the application.
- **Deformed items**: The impact of the Scaling DSDI was measured as the percentage of items deformed in a non-uniform manner.

6 Results

The results are summarised for each application, for each shape and phase, in Tables 1 and 2. The phases used in the tables represent the following stages of the execution of the technique as follows:

- **Phase 0**: When the application is first started, before execution of the technique starts.
- **Phase 1**: After the display shape borders are applied to the application.
- **Phase 2**: After the virtual projection stage of the technique is carried out.
- **Phase 3**: After the position pulling stage of the technique is completed.

6.1 Grid application

Table 1 shows that the technique reduced the initial occlusion for all shapes without incurring any occlusion from overlapping items or deformation of items throughout its execution. However, the virtual projection stage did increase the change in scale and the proportion of the display shape which went unused. The position pulling stage of the technique did reduce the amount of the display shape which went unused quite significantly for some shapes but did increase the change in layout. The change in layout for some display shapes was quite significant; for example, see the triangle in Fig. 8, where the grid layout becomes nearly unrecognisable.

Table 1 DSDI measurements from the study in *SynergyNet*'s *Grid* application

Grid application							
		DSDI measurements					
Shape	Phase	Occluded items	Overlapping items	Change in layout	Display shape unused	Change in scale	Deformed items
Rectangle	0	0%	0%	0%	0%	0%	0%
Circle	1	66.7%	0%	0%	0%	0%	0%
	2	0%	0%	0%	35.8%	39.5%	0%
	3	0%	0%	6.4%	20.9%	39.5%	0%
Triangle	1	77.8%	0%	0%	0%	0%	0%
	2	0%	0%	0%	50.1%	48.8%	0%
	3	0%	0%	20.7%	7.6%	48.8%	0%
Intersecting circles	1	66.7%	0%	0%	0%	0%	0%
	2	0%	0%	0%	56.9%	44.2%	0%
	3	0%	0%	27.4%	14.6%	44.2%	0%
Semi-circle	1	88.9%	0%	0%	0%	0%	0%
	2	0%	0%	0%	40.9%	44.2%	0%
	3	0%	0%	11.9%	13%	44.2%	0%

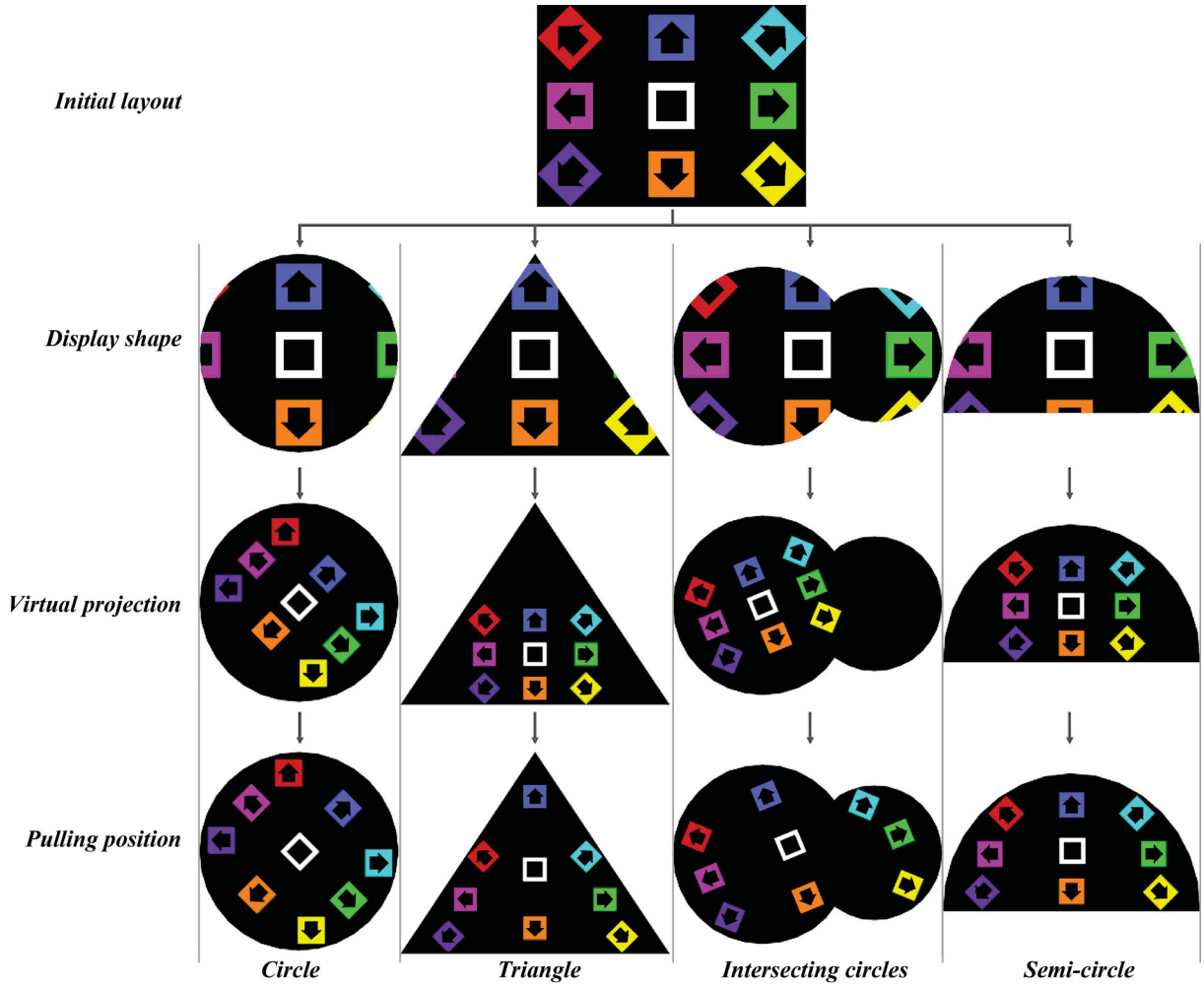

Fig. 8 Influence of our technique on *SynergyNet*'s *Grid* application.

6.2 Simple Map application

In the map application, the large item displaying the map was exempted from the influence of the position pulling stage of the technique.

Table 2 shows that the technique reduced the initial occlusion for all shapes without incurring any additional occlusion from overlapping items or deformation of items throughout its execution. However, the virtual projection stage increased the change in scale and the amount of the display shape which went unused. The position pulling stage of the technique reduced the amount of the display shape which went unused quite significantly for some shapes but also increased the change in layout. The change in layout for some of the display shapes was quite significant, as for the grid app, implying that the display shape has a large influence on the impact of the technique, not just the content.

Figure 9 demonstrates the importance of the layout of content items. Up to the virtual projection stage of the technique, the flag items are correctly aligned with the countries on the map. However, the position pulling stage deforms the layout of the flags in such a way that disengages this alignment. If it is important for this alignment to be kept then the flags should also be exempted from the position pulling stage of the technique; as a consequence, however, areas of the display will be left unused by the layout. There is a clear choice developers must make between preserving the layout of content items or maximising usage of the display's real estate. A clear example of this choice can be seen in the technique's influence on the *intersecting circles* display shape in Fig. 9. Without the influence of the position pulling stage of the technique, a large region of the display is left unused by the initial layout, resulting in the *Display Use*

Table 2 DSDI measurements from the study in *SynergyNet*'s *Simple Map* application

Simple Map application							
		DSDI measurements					
Shape	Phase	Occluded items	Overlapping items	Change in layout	Display shape unused	Change in scale	Deformed items
Rectangle	0	0%	20%	0%	0%	0%	0%
Circle	1	60%	20%	0%	0%	0%	0%
	2	0%	20%	0%	37.9%	42%	0%
	3	0%	20%	5.2%	37.1%	42%	0%
Triangle	1	100%	20%	0%	0%	0%	0%
	2	0%	20%	0%	50.2%	50%	0%
	3	0%	20%	17.6%	22.7%	50%	0%
Intersecting circles	1	80%	20%	0%	0%	0%	0%
	2	0%	20%	0%	58.1%	48.2%	0%
	3	0%	20%	16.4%	42%	48.2%	0%
Semi-circle	1	80%	20%	0%	0%	0%	0%
	2	0%	20%	0%	40.3%	44.6%	0%
	3	0%	20%	9.9%	28.2%	44.6%	0%

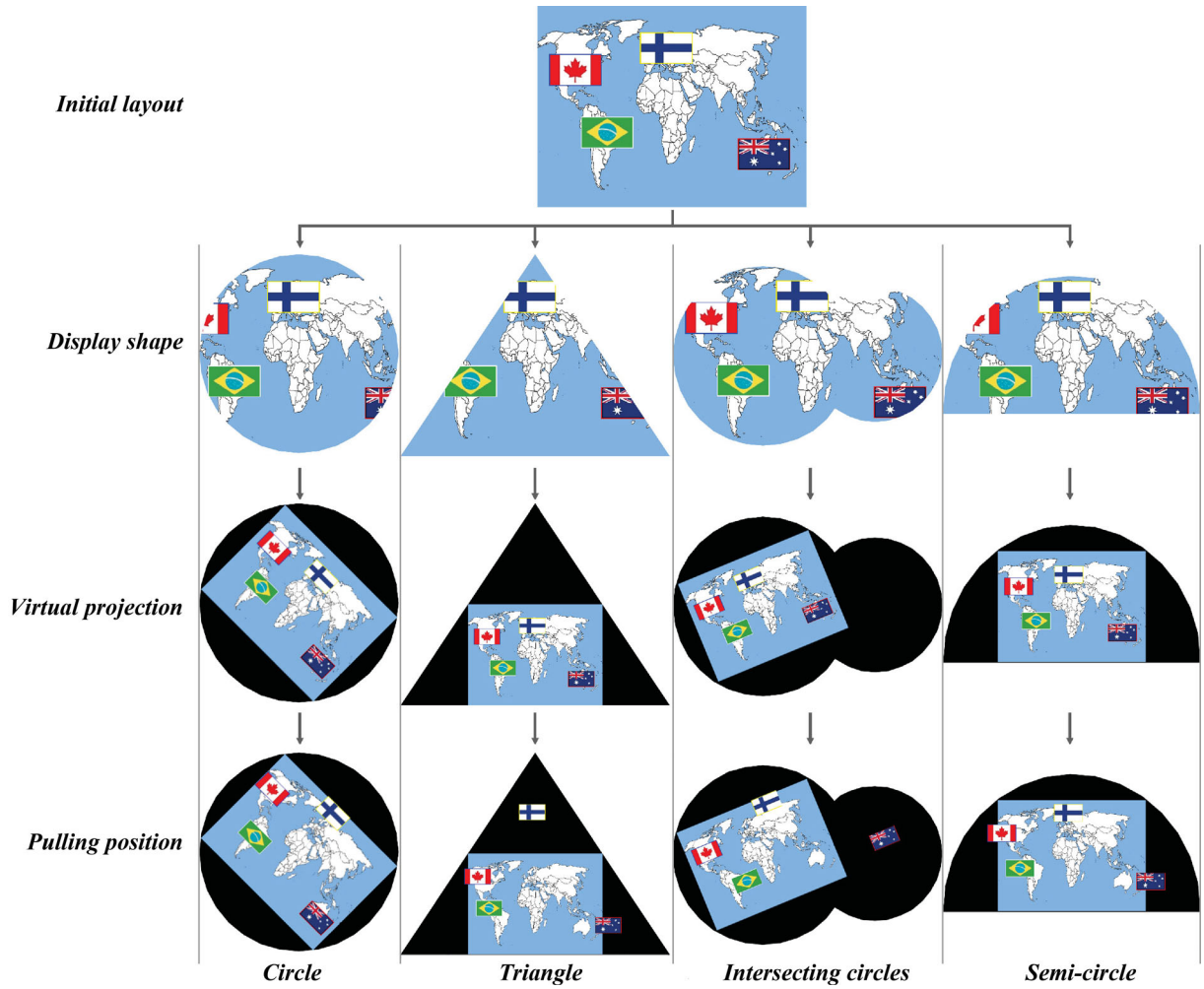

Fig. 9 Influence of the implemented technique on *SynergyNet*'s *Simple Map* application.

DSDI occurring. However, with the influence of the position pulling stage of the technique, several of flag items appear off the map, away from their respective positions, indicating the *Layout DSDI* has occurred.

7 Conclusions and future work

Developers must consider how any proposed method counters the DSDIs. The identification, or creation of, an adequate DSDI-minimising technique which allows a particular system to adapt to different display shapes could form part of design guidelines for emerging technologies. It is important to note that the list of DSDIs for a system should not be the only influence on the design of a DSDI-minimising technique; the need for applications to control the magnitude of a DSDI-minimising technique's influence is also important.

The development of the technique presented in Section 4 has shown that when designing software for use on differently shaped tabletop displays, the *Cropping DSDI* must be considered. In addition to this, secondary DSDIs which can result from attempts to resolve the initial cropping must also be addressed. Developers should also consider which DSDIs their software can tolerate and which must be handled by a DSDI-minimising technique. Choices must be made between preserving the layout of content items and maximising display usage.

In the study, both stages of the technique were applied to investigate its impact; in typical usage, an application developer using the framework would have the option to override them for specific content items. For example, they could apply scale limits to the virtual projection for all content in an application or to individual items. The ability to choose which stages of the technique affect each content item gives developers control over their influence. Without this ability there is still a danger of unacceptable DSDIs occurring in software despite the presence of the implemented technique. This highlights that any technique intended to minimise the influence of the potential DSDIs in a system should not only be adaptable to different display shapes, but also be adaptable to the needs of the developer. Doing so allows developers to make informed, user-centered design and implementation decisions on how content should be displayed.

However, developers must consider the limitations of the technique before adopting it. One such limitation relates to items providing backgrounds. Depending on what stages of the technique are applied to a background item the result may either be a partially occluded background or blank spaces left where the item is absent due to it being scaled to fit the maximal rectangle. Both of these outcomes may be undesirable in some scenarios. Possible solutions which could be explored include adding an extra step to apply warping to the background item or context-aware filling of blank regions of the display. Another limitation is the assumption that there is no hardware-level attempt to fit the output to the display shape.

Although the technique presented in this work helps to overcome the DSDIs of tabletop interfaces, it is not always guaranteed to be the best solution. It may be possible in many combinations of content and display shapes to find solutions where the impact of the steps taken to avoid DSDIs is lower than the impact of the presented technique. A solution of lower impact is generally preferable for developers, as it allows content to appear more consistent across different displays, allowing the software to appear more intuitive and familiar to users across devices. Identifying what solution is best suited for a given situation should be at the centre of any future research involving this topic. Despite lack of a guarantee that the technique is optimal for a given situation, it is still a significant step forward for resolving DSDIs as it can (i) always supply an adequate solution in a wide range of scenarios; (ii) be automated as part of an implementation as shown in this work; and (iii) allow developers some control over its execution.

There are various reasons why it is important that software should become display shape independent. There is now a growing push for software systems to become ubiquitous with the increased availability of natural user interfaces; for a system to become ubiquitous it must fit its environment [8]. Being able to change the display shape of a system would aid in achieving this.

The examples of non-rectangular displays discussed in Section 2 reinforce the need for their support. In addition to these, many devices coming to the mass consumer market—for example, recently popular wearable devices—utilise non-rectangular displays, especially smart watches [35]. Figure 10 shows a prototype built by a leasing furniture manufacturer

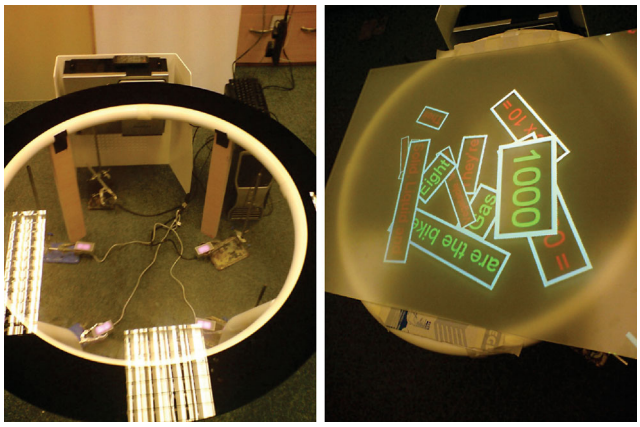

Fig. 10 A prototype of a circular tabletop display.

featuring a circular tabletop interface intended for use as an interactive kiosk for public spaces. The lack of readily available software for the interface's circular display meant that to use the tabletop, bespoke software would need to be commissioned at a significant cost to the manufacturer. Readily available off-the-shelf products intended for interactive kiosks could not be directly used. This highlights the need to make it as simple as possible for software to support different display shapes.

The technique presented in this paper can be adopted by developers to allow their content to fit different display shapes, much like the guidelines presented in the work of Serrano et al. [13, 14]. However, while our technique deviates from their guidelines, it provides developers with greater freedom and flexibility. Without encumbering software developers and content designers with additional restrictions, the technique is able to resolve the DSDIs with minimal input. This saves development time when adding support for different display shapes and also offers more freedom in the design of content.

Furthermore, the technique outlined in Section 4 could also be used beyond tabletop systems. However, the DSDIs in other systems may differ from those presented in this work [14]. For example, attempts to resolve the initial *Cropping DSDI* may require content items to be rotated. If a user views the display from a fixed viewpoint, e.g., using a vertical display, then any change in content orientation is undesirable: rotated visual content items, such as text, could be difficult to understand. This is not an issue for systems which utilise horizontal interfaces and allow users to adopt multiple positions around the display. Future work could entail the outlining of DSDI for other kinds of systems and the production of techniques to minimise their influence.

Further research involving the technique could involve investigating how it functions with more extreme shapes. Although the shapes used in the study vary somewhat and cover a wide range of what may be used as tabletop display shapes, it is possible that shapes varying from rectangular display in much more extreme ways could be useful. Research utilising these extreme shapes could involve finding acceptable limits to place on parts of the technique such as maximum distances to pull content items.

While this work details an implementation of the technique, the next step should be a full user study investigating how well the technique works for both real-world developers and end users; for example, a study similar to that carried out by Serrano et al. [14] to review the impact of their proposed guidelines is likely to be a suitable approach. By instructing developers to utilise the technique when building software, it should be possible to discover the suitability of the technique in real-world scenarios. We can then review the impact of the technique in these implementations to find if the results are suitable for end-users.

It is important to note that the scope of this work entailed resolving DSDIs relating to the layout of visual contents on tabletop interfaces. Future work has the potential to apply the technique presented to other kinds of interfaces. This future work would likely focus on additional restrictions the technique would have to take into account, such as the need for a specific orientation with vertical displays. Future work could also consider expanding the technique to resolve more than just DSDIs relating to the layout of visual content. Extra steps or adding limits to the transformations of elements within the technique may help improve usability and readability when applying content to a different display shape.

The technique presented in this paper can be used in various scenarios; one such use is its implementation in the application layer of a piece of software; this would allow a specific layout of visual content items to be dynamically adapted to different display shapes. As this implementation of the technique would be customised to the application's specific layout, it would ensure the best results. However, this does have the drawback that the

technique would need to be redefined for every unique application.

Another use for the technique is its implementation in an adaptable and extensible framework. In a framework-level implementation, the same technique could be employed by various applications to adapt their visual contents to different display shapes; this is similar to how systems like SUPPLE [16] adapt contents to display parameters. However, a drawback to this approach is that the technique's implementation may be unsuitable for some applications. For example, if the implementation does not enforce scale limits, the *Scaling DSDI* may occur. This may be acceptable for some applications, such as those intended to use content which can still communicate information to the user at any size. However, it would be unacceptable for other applications, such as those using text. Therefore, it is important when implementing this technique at a framework level that applications can change how the technique influences them. The technique's use of higher-level functions, such as the ray firing used in the position pulling stage, make it unsuitable for implementation in lower level software.

The technique could also be implemented as part of a design tool used to adapt the layout of visual content, similar to GUMMY [15]. Using this implementation, software developers could use the resulting tool to adapt a layout of content items to a specific display shape. Developers could then make any further changes manually if the tool's automatically generated DSDI-minimising technique were not adequate. Once a developer is satisfied with the software layout they could then specify its use in their software when the appropriate display shape is present. This approach has the advantage of allowing developers to check that the layout will be suitable for a specific shape. However, it does potentially limit the display shapes the system can use to those the developer has input into the tool.

Finally, the technique could be transformed into a set of guidelines for developers to follow when creating software interfaces without the use of a design tool. The guidelines could also incorporate the decisions to be made concerning the trade-offs in which DSDIs are minimised. These guidelines could then be formalised [36], allowing their application to software to be automated in places. This would aid with the technique's integration into frameworks and design tools. With the technique implemented in a tool used to design a wider range of software, its influence can be tested on a wider range of user interfaces with varying complexities in future studies, especially in educational contexts [31, 37].

In summary, the technique presented in this paper can efficiently minimise the effects of the identified DSDIs, but requires trade-offs. Developers using the technique must decide which DSDIs cannot be tolerated by their software and must adapt the technique according. Future tabletop software development would benefit greatly from the creation of guidelines to follow when adapting software to become display shape independent. The technique produced in this paper, or another similar DSDI-minimising technique, could form part of these guidelines.

Acknowledgements

This work was partially funded under the UK's EPSRC/ERSC Teaching and Learning Research Programme (TLRP) *SynergyNet* project (RES-139-25-0400). The authors would also like to thank Professor Liz Burd and Dr. Andrew Hatch supervising the primary author's master degree from which this work originally stems. The authors would also like to thank the members of the Durham University Technology Enhanced Learning Special Interest Group for supporting the redrafting of this manuscript. Source code for the technique's implementation is available at https://github.com/synergynet/synergynet2.1.

References

[1] Dietz, P.; Raskar, R.; Booth, S.; van Baar, J.; Wittenburg, K.; Knep, B. Multi-projectors and implicit interaction in persuasive public displays. In: Proceedings of the Working Conference on Advanced Visual Interfaces, 209–217, 2004.

[2] Boyd, J. Circular LCD debuts. 2007. Available at http://spectrum.ieee.org/computing/hardware/circular-lcd-debuts.

[3] Finney, M. D.; Oliver, M. W.; Pierce, P. M.; Sutherland, T. Communication device. U.S. Patent USD600228. 2009.

[4] Battersby, S. J. Non rectangular display device. U.S. Patent 7,253,865. 2007.

[5] Hansen, T. E.; Hourcade, J. P.; Virbel, M.; Patali, S.; Serra, T. PyMT: A post-WIMP multi-touch user interface toolkit. In: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, 17–24, 2009.

[6] Shen, C.; Vernier, F. D.; Forlines, C.; Ringel, M. DiamondSpin: An extensible toolkit for around-the-table interaction. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 167–174, 2004.

[7] Weiser, M. The computer for the 21st century. *Mobile Computing and Communications Review* Vol. 3, No. 3, 3–11, 1999.

[8] Greenfield, A. *Everyware: The Dawning Age of Ubiquitous Computing*. Peachpit Press, 2006.

[9] Mostafa, M.; Crick, T.; Calderon, A. C.; Oatley, G. Incorporating emotion and personality-based analysis in user-centered modelling. In: *Research and Development in Intelligent Systems XXXIII*. Bramer, M.; Petridis, M. Eds. Springer Cham, 383–389, 2016.

[10] Smith, S. P.; Burd, E.; Rick, J. Developing, evaluating and deploying multi-touch systems. *International Journal of Human-Computer Studies* Vol. 70, No. 10, 653–656, 2012.

[11] Vernier, F.; Lesh, N.; Shen, C. Visualization techniques for circular tabletop interfaces. In: Proceedings of the Working Conference on Advanced Visual Interfaces, 257–265, 2002.

[12] Kitchenham, B. Procedures for performing systematic reviews. Technical Report TR/SE-0401. Keele University, 2004.

[13] Serrano, M.; Roudaut, A.; Irani, P. Investigating text legibility on non-rectangular displays. In: Proceedings of the CHI Conference on Human Factors in Computing Systems, 498–508, 2016.

[14] Serrano, M.; Roudaut, A.; Irani, P. Visual composition of graphical elements on non-rectangular displays. In: Proceedings of the CHI Conference on Human Factors in Computing Systems, 4405–4416, 2017.

[15] Meskens, J.; Vermeulen, J.; Luyten, K.; Coninx, K. Gummy for multi-platform user interface designs: Shape me, multiply me, fix me, use me. In: Proceedings of the Working Conference on Advanced Visual Interfaces, 233–240, 2008.

[16] Gajos, K.; Weld, D. S. SUPPLE: Automatically generating user interfaces. In: Proceedings of the 9th International Conference on Intelligent User Interfaces, 93–100, 2004.

[17] Waldner, M.; Grasset, R.; Steinberger, M.; Schmalstieg, D. Display-adaptive window management for irregular surfaces. In: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, 222–231, 2011.

[18] Constantine, L. L.; Lockwood, L. A. D. Software for use: A practical guide to the models and methods of usage-centered design. *ACM SIGCHI Bulletin* Vol. 32, No. 1, 111–114, 1999.

[19] McNaughton, J.; Crick, T.; Hatch, A. Determining device position through minimal user input. *Human-centric Computing and Information Sciences* Vol. 7, No. 1, 37, 2017.

[20] Milliron, T.; Jensen, R. J.; Barzel, R.; Finkelstein, A. A framework for geometric warps and deformations. *ACM Transactions on Graphics* Vol. 21, No. 1, 20–51, 2002.

[21] Shen, C.; Ryall, K.; Forlines, C.; Esenther, A.; Vernier, F. D.; Everitt, K.; Wu, M.; Wigdor, D.; Morris, M. R.; Hancock, M.; Tse, E. Informing the design of direct-touch tabletops. *IEEE Computer Graphics and Applications* Vol. 26, No. 5, 36–46, 2006.

[22] Scott, S. D.; Sheelagh, M.; Carpendale, T.; Inkpen, K. M. Territoriality in collaborative tabletop workspaces. In: Proceedings of the ACM Conference on Computer-Supported Cooperative Work, 294–303, 2004.

[23] Smith, S. P.; Burd, E. L.; Ma, L.; AlAgha, I.; Hatch, A. Relative and absolute mappings for rotating remote 3D objects on multi-touch tabletops. In: Proceedings of the 25th BCS Conference on Human-Computer Interaction, 299–308, 2011.

[24] Schöning, J.; Brandl, P.; Daiber, F.; Echtler, F.; Hilliges, O.; Hook, J.; Löchtefeld, M.; Motamedi, N.; Muller, L.; Olivier, P.; Roth, T.; von Zadow, U. Multi-touch surfaces: A technical guide. Technical Report TUM-I0833. University of Munich, 2008.

[25] Aggarwal, A.; Suri, S. Fast algorithms for computing the largest empty rectangle. In: Proceedings of the 3rd Annual Symposium on Computational Geometry, 278–290, 1987.

[26] Naamad, A.; Lee, D. T. On the maximum empty rectangle problem. *Discrete Applied Mathematics* Vol. 8, No. 3, 267–277, 1984.

[27] Toussaint, G. T. Computing largest empty circles with location constraints. *International Journal of Computer & Information Sciences* Vol. 12, No. 5, 347–358, 1983.

[28] Cotting, D.; Gross, M. Interactive environment-aware display bubbles. In: Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology, 245–254, 2006.

[29] Raskar, R.; van Baar, J.; Beardsley, P.; Willwacher, T.; Rao, S.; Forlines, C. iLamps: Geometrically aware and self-configuring projectors. *ACM Transactions on Graphics* Vol. 22, No. 3, 809–818, 2003.

[30] Van Dam, A. User interfaces: Disappearing, dissolving, and evolving. *Communications of the ACM* Vol. 44, No. 3, 50–52, 2001.

[31] McNaughton, J.; Crick, T.; Joyce-Gibbons, A.; Beauchamp, G.; Young, N.; Tan, E. Facilitating collaborative learning between two primary schools using large multi-touch devices. *Journal of Computers in Education* Vol. 4, No. 3, 307–320, 2017.

[32] AlAgha, I.; Hatch, A.; Ma, L.; Burd, E. Towards a teacher-centric approach for multi-touch surfaces in classrooms. In: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, 187–196, 2010.

[33] Kaltenbrunner, M.; Bencina, R. reacTIVision: A computer-vision framework for table-based tangible interaction. In: Proceedings of the 1st International Conference on Tangible and Embedded Interaction, 69–74, 2007.

[34] Higgins, S.; Mercier, E.; Burd, L.; Joyce-Gibbons, A. Multi-touch tables and collaborative learning. *British Journal of Educational Technology* Vol. 43, No. 6, 1041–1054, 2012.

[35] Jung, Y.; Kim, S.; Choi, B. Consumer valuation of the wearables: The case of smartwatches. *Computers in Human Behavior* Vol. 63, 899–905, 2016.

[36] Ngo, D. C. L.; Teo, L. S.; Byrne, J. G. Formalising guidelines for the design of screen layouts. *Displays* Vol. 21, No. 1, 3–15, 2000.

[37] Joyce-Gibbons, A.; McNaughton, J.; Tan, E.; Young, N.; Beauchamp, G.; Crick, T. SynergyNet into schools: Facilitating remote inter-group collaborative learning using multi-touch tables. In: Proceedings of the 12th International Conference on Computer Support Collaborative Learning, 2017.

James McNaughton is a researcher at Durham University, UK, whose research interests include HCI, natural user interfaces, and augmented reality. His current work involves investigating the use of emerging interaction technologies in classroom environments.

Tom Crick is a professor of digital education & policy at Swansea University, UK. His research interests are interdisciplinary, including data science, intelligent systems, digital public services, software sustainability, and computer science education.

Shamus Smith is a senior lecturer in computer science at the University of Newcastle, Australia. His current research interests include touch-based technologies, mobile technology for augmented reality, and the reuse of gaming technology.